IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In RE application of: James C. Fye	Group Art Unit: 2421
Serial No.: 10/699,311	Examiner: Chenea Smith
Filed: October 30, 2003	Confirmation No.: 3928

For: ARCHITECTURE FOR MULTI-CHANNEL VIDEO PROCESSING

Attorney Docket No.: H0005246 (256.155US1)

DECLARATION OF JAMES C. FYE UNDER 37 C.F.R. § 1.131

I, James C. Fye, state that:

1. I am currently employed by Honeywell International, Inc. ("Honeywell") of Morristown, New Jersey.

2. I am the named inventor of U.S. Patent Application Serial No. 10,699,311 entitled "Architecture for Multi-Channel Video Processing" (hereinafter "the Present Application"), which was filed on October 30, 2003.

3. The Present Application was assigned to Honeywell, which assignment was recorded on Reel/Frame 014673/0570 on October 30, 2003.

4. U.S. Patent Application Publication No. 2005/0028225 (i.e., the reference being cited by the Patent Office) was published on February 3, 2005 and filed on July 29, 2003, which is the earliest effective date of U.S. Patent Application Publication No. 2005/0028225 under 35 U.S.C. 103(a).

5. I conceived and actually reduced to practice the invention defined by claims 1-28 in the Present Application before the earliest effective date of U.S. Patent Application Publication No. 2005/0028225. As evidence of the actual reduction to practice of the invention defined by claims 1-28 of the Present Application, a copy of an invention disclosure that I authored and submitted to Honeywell prior to the earliest effective date

of U.S. Patent Application Publication No. 2005/0028225 is enclosed herewith as Appendix A, which invention disclosure indicates that I had actually reduced the invention to practice (see page 2, item 5(a)) prior to July 29, 2003.

6. As further evidence of actual reduction to practice, enclosed as Appendix B is a copy of a portion of the top level VHDL file for an FPGA, which copy shows various modifications I made to the VHDL file until a first prototype for the invention was completed. Specifically, page 4 of Appendix B shows the final modifications made to the VHDL file for first prototype.

7. All of the dates that have been redacted in the invention disclosure included in Appendix A and the VHDL file included in Appendix B are prior to the earliest effective date of U.S. Patent Application Publication No. 2005/0028225 (i.e., July 29, 2003).

8. Other redacted information in the invention disclosure included in Appendix A relates to my personal information.

9. I hereby declare that all statements made herein are of my own knowledge, are true, and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under § 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application, any patent issuing thereon, or any patent to which this verified statement is directed.

Executed on 8/17/09

[illegible]

James C. Fye

APPENDIX A

Honeywell CONFIDENTIAL
ATTORNEY-CLIENT PRIVILEGED

Invention Record (Docket) No.: H0005246

Origin Date: [redacted]

SBE: 1623 - Phnx., AZ - BUSINESS/COMMUTER AVIATION SYST

Attorney(s): Jackson, Miriam -

File Location: PH - Phoenix, AZ

Title: A Scalable Method to Robustly Display Multiple Channels of Real Time Synchronized or Unsynchronized Analog Video Information at up to Full Video Frame Rate Performance on a RGB Graphics Display

---

Inventor: Fye, James C
Address: [redacted]
[redacted]
Phone: [redacted] Fax:
[redacted]
Citizenship: USA

SSN: [redacted]
County: [redacted]
Supervisor: Mike Coyle

---

1. Briefly describe the technical or commercial problem or need that this invention is intended to solve.
With a growing need for real time situational awareness on aircraft (for safety, security, enhanced vision, moving maps, etc.), comes a growing need for robust display of multiple simultaneous analog video channels. Typically, a system that renders multiple simultaneous analog video channels in real time will simply implement an analog video multiplexer at the front end of the video-processing pipeline. A system of this nature has a "divide-by-n" type of degraded frame rate performance and gets quite slow when four or more simultaneous analog video channels are desired to be displayed. A synchronized "n" channel analog video rendering system of this type (displaying "n" video channels simultaneously) will do so with a degraded frame update rate of "30Hz/n" for the NTSC analog video format standard (60Hz interlaced fields, 30Hz frame updates) and "25Hz/n" for the PAL/SECAM analog video format standard (50Hz interlaced fields, 25Hz frame updates). An unsynchronized "n" channel analog video rendering system of this type can be two to three times worse than the synchronized system when taking into account the (dead) time needed for the video decoder to lock on to the incoming analog video signal. A four-channel synchronized NTSC system of this type would produce 7.5Hz frame updates while an unsynchronized NTSC system of this type would produce around 3Hz frame updates. An eight-channel NTSC system of this type is even more dramatic in degraded frame rate performance producing 3.75Hz frame updates (synchronized) and around 1.5Hz frame updates (unsynchronized). This invention addresses this "divide-by-n" type of degraded performance by providing a scalable method that can recover up to the full frame rate performance level for multiple channels of simultaneously displayed video in real time. It also handles in real time scenarios associated with failed and 'now' passing video sources for robust situational awareness.

---

2. Briefly describe how this invention solves the problem or meets the need.
This invention creates a robust real time video rendering environment with the following features: -Improved multi-channel simultanously displayed video frame update rates by using dedicated video decoders (recommended), a non-blocking switch network, and multiple video processing pipelines (qty scaled as desired) fed into a write interleaving multiplexer (bandwidth scaled as desired) to render to the RGB display system. -Supports prompt detection and display of individual failed video sources along with a method to promptly recover a 'now' passing video source. -Supports further improvements in update rates of unsynchronized video source systems by using video-scaling-field-processing for image sizes that will support that method in real time.

---

3. Describe how to make and use the invention. Please indicate which embodiment(s) are preferred and describe the best way known to you to practice the invention. Attach relevant documents. (If the invention is a device or process, please provide a drawing or flow chart.) (If you are unfamiliar with the contents and preparation of a patent application, please refer to the Guidelines for the Preparation of Invention Disclosures.

See attachment.
Document(s):
H0005246_MU1_NchannelRTvideo.doc

4(a). To the best of your recollection what is the earliest date on which the invention was conceived? Who conceived the invention? Attach documents which evidence the foregoing.

Conception Date: [redacted] Who conceived it?: **James Fye**
Document(s):
H0005246_CD1_ei243773.doc

4(b). Is there a non-inventor who witnessed the conception? If so, please identify him/her and attach any documents which evidence the witnessing.
No
Witness Name: Witness Phone: Document(s):

5(a). To the best of your recollection, what is the earliest date on which the invention was reduced to practice (i.e. made)? Who reduced the invention to practice. Attach documents which evidence the foregoing. If no reduction to practice, type "n/a".
First Practice Date: [redacted] Who reduced it to practice?: **New Video Module prototypes**
Document(s):

5(b). Is there a non-inventor who corroborated the reduction to practice? If so, please identify him/her, the corroborating activity (i.e., over-the-shoulder corroboration or repeating the experiment), and the date of the activity. Attach documents which evidence the foregoing.

Non-inventor corroborator?:	First Corroborator Name:	First Corroborator Phone:
First Practice Corroboration Date:	First Practice Corroborator Activity: **TBD**	

Document(s) related to corroboration event:

5(c). For each example of the invention and each comparative example on which you intend to rely in the patent application, please indicate when the example was generated, who conducted the experiment and where this example is recorded (e.g., volume, page and author or laboratory notebook) and attach a copy of these records. If no example available, type "n/a".
Example(s):
Who conducted the experiment?: **TBD**
Where is example recorded?: Document(s):

6(a). Did this invention arise in a program that is funded in whole or part by the U.S. Government or another company, or any entity other than Honeywell?

6(b). If so, please identify the program (including government contract number, if applicable) and the entity sponsoring the program and provide a copy of any agreement between the parties concerning the program.

Outside Funding Program:

Contract Number *(if applicable)*:
Outside Funding Entity:
Document(s) related to funding agreement:

7(a). To your knowledge, is this invention subject to any agreement between Honeywell and a third party (e.g., a secrecy agreement, license agreement, joint development agreement, etc.)?
No

7(b). If so, please identify the agreement and the other party and attach a copy of the agreement if one is available.
Third party agreement ID:
Third party name:
Document(s) related to any third party agreement:

---

8. You have a duty to disclose to the U.S. Patent and Trademark Office all relevant prior art of which you are aware. Please list all such prior art (e.g., patents, publications, brochures, Honeywell and third-party products) known to you. If a prior art search has been conducted, it must be included. Briefly indicate how this invention is different from the prior art. See 1 and 2 above.
List of prior art:

How invention is different from the prior art:

9(a). Has the product or process which is the subject of this invention disclosure been disclosed, sold or offered for sale to anyone outside of Honeywell or to the general public.
No

9(b). If so, when and to whom was it disclosed, sold or offered for sale? If it was disclosed, was a secrecy agreement in place? Attach documents which evidence the sale or offer for sale.

Date it was disclosed:
Whom disclosed to:
Disclosure Sales Agreement?:
Document(s) which evidence the sale or offer for sale:

9(c). Does the business intend to disclose, sell or offer to sell the invention to anyone outside of Honeywell or to the general public in the near future? If so, to whom and when is this disclosure, sale or offer for sale planned?

For whom are future sales planned:
Date future sale is planned:

10(a). Does this invention relate to any other: (i) issued patents, (ii) pending patent applications, or (iii) previously submitted invention disclosures, of Honeywell?
No

10(b). If so, please identify the related matter and indicate whether this is an improvement on an earlier invention: Other patents related matter is:

Is this an improvement?:

---

11. Please specify the product(s) to which this invention disclosure relates.

Primus Epic Video Module

12. Please indicate keywords for identifying this invention disclosure.
Video Rendering

Only one witness signature is required.

Witness Name: Mike Coyle	Inventor Name: JAMES C. FYE
Witness Signature: M. D. Coyle	Inventor Signature: [illegible]
Date: [redacted]	Date: [redacted]
Inventor Name:	Inventor Name:
Inventor Signature:	Inventor Signature:
Date:	Date:
Inventor Name:	Inventor Name:
Inventor Signature:	Inventor Signature:
Date:	Date:
Inventor Name:	Inventor Name:
Inventor Signature:	Inventor Signature:
Date:	Date:

Send to:
Miriam - Jackson

,

A Scalable Method to Robustly Display Multiple Channels of Real Time Synchronized or Unsynchronized Analog Video Information at up to Full Video Frame Rate Performance on a RGB Graphics Display

Jim Fye
[redacted]

Background

With a growing need for real time situational awareness on aircraft (for safety, security, enhanced vision, moving maps, etc.), comes a growing need for robust display of multiple simultaneous analog video channels. Typically, a system that renders multiple simultaneous analog video channels in real time will simply implement an analog video multiplexer at the front end of the video-processing pipeline. A system of this nature has a "divide-by-n" type of degraded frame rate performance and gets quite slow when four or more simultaneous analog video channels are desired to be displayed. A synchronized "n" channel analog video rendering system of this type (displaying "n" video channels simultaneously) will do so with a degraded frame update rate of "30Hz/n" for the NTSC analog video format standard (60Hz interlaced fields, 30Hz frame updates) and "25Hz/n" for the PAL/SECAM analog video format standard (50Hz interlaced fields, 25Hz frame updates). An unsynchronized "n" channel analog video rendering system of this type can be two to three times worse than the synchronized system when taking into account the (dead) time needed for the video decoder to lock on to the incoming analog video signal. A four-channel synchronized NTSC system of this type would produce 7.5Hz frame updates while an unsynchronized NTSC system of this type would produce around 3Hz frame updates. An eight-channel NTSC system of this type is even more dramatic in degraded frame rate performance producing 3.75Hz frame updates (synchronized) and around 1.5Hz frame updates (unsynchronized).

Robust Multi-Channel Real Time Video – Frame Update Rate

Acceptable degraded video frame update rate performance is not an exact science. It will vary widely between individuals and it also has dependence on the actual video information content. For example, looking at a somewhat static image (like a moving map) would have less degraded video frame update rate performance issues than a fast moving image (like enjoying a DVD movie). None the less, an attempt will be made to put into perspective video frame update rate performance levels:

- 30Hz video frame updates: good-full frame update rate performance (NTSC)
- 15Hz video frame updates: ok-occasionally may appear jerky
- 10Hz video frame updates: ok-but jerky
- 7.5Hz frame updates: marginal-jerky
- 3.75Hz frame updates: obnoxiously slow-jerky

Ideally a scaleable design approach would be able to easily trade-off the costs associated with implementing the preferred full frame rate performance of rendering simultaneously

[illegible]
[redacted]

multiple channels of real time analog video versus not. For decent performance, staying above a 10Hz frame update rate level of performance is very desirable.

Robust Multi-Channel Real Time Video – Real Time Video Selection and Sizing

The ability to flexibly select a video view port window size on the display as well as the selection of video sources to be viewable in that video view port window is an important aspect of a robust multi-channel real time video system. The video view port window size selection process should be able to range from various partial-screen selection options to a full-screen size selection option. The selection of the currently viewable number of sources to be displayed within that view port window should be one to any combination-of to all-of the video channels available. To support this flexibility a scaler is necessary to scale the incoming video stream to the correct 4:3 format necessary to fit the desired number of video sources into the desired video view port display area.

From the video processing perspective, when a new selection is requested (view port size, sources viewable) an individual image location/size calculation is needed for each video source currently selected so that the video rendering process can correctly locate each image on the display. Once that information is derived, the real time video rendering process can begin. The video rendering process uses the individual image location/size information to ensure that each image is rendered only in its allowable area. That is, enforced containment of the expected number of individual video image pixels and lines ensures that adjacent video image corruption will not occur during the multi-channel real time video rendering process.

Robust Multi-Channel Real Time Video – Failed Video Source Conditions

When dealing with real time situational awareness information a failed video source should be promptly flagged. An unflagged frozen image is very undesirable. In addition, when rendering multiple channels of real time video simultaneously the failed video sources should have minimal impact (ideally no impact) on the valid video sources still available. A 'no impact failed video source design' would have no degradation in frame update rate performance on the remaining valid video sources when the failed video timeout monitor is in parallel with the overall in-time processing path. A 'minimal impact failed video source design' would allow a small degradation in frame update rate performance on the remaining valid video sources when the time needed to trip a failed video timeout monitor is serially in the overall in-time processing path. Some flagging failed video display examples are:

- Retain last image with descriptive text overlay (i.e. "VIDEO FAIL")
- Blank/Black failed video source display area
- Blank/Black failed video source display area and display descriptive text overlay (i.e. "VIDEO2 FAIL")
- Blank/Black failed video source display area and overlay an "X" over the whole video source display area.
- Blank/Black failed video source display area and overlay an "X" over the whole video source display area along the with a descriptive text overlay as well (i.e. "VIDEO4 FAIL").

If a selected failed video source should become valid, it should be promptly re-displayed.

[illegible]
[redacted]

Robust Multi-Channel Real Time Video – A Scalable Design Method

Capturing the robust multi-channel real time video thoughts from above, a scalable design method, which meets those desires, will now be presented. The scalability tradeoffs of this design approach will trade-off the video processing pipeline logic gate count utilization and video frame buffer memory bandwidth needed vs. the desired video frame update rate performance level. This design can scale to support full frame rate performance while simultaneously displaying all incoming video channels.

Since unsynchronized video systems are the general rule rather than the exception, it is highly recommended that the (dead) time associated with the video decoder to lock on to the incoming analog video signal be eliminated. Video decoder lock (dead) times in a real time rendering environment are relatively large. Video decoder lock (dead) times can easily span multiple video fields (16.667ms/20ms) and sometimes even multiple video frames (33.333ms/40ms)! To eliminate this significant dead time in the overall in-time processing path, it is highly recommended that the design supports a video decoder per analog video input channel. This scalable design approach can be made to work fewer video decoders (with an addition of analog video multiplexer arrangement) but at least two video decoders are needed to improve upon the traditional "divide-by-n" type of simultaneous multi-channel real time video rendering to a display.

The attached block diagram shows "n" analog video inputs being process by "n" video decoders and "p" video processing pipelines. The scaler part of the video processing path may be included with the decoder or within the video processing pipeline itself. Here it is shown as being with the video decoder. The particulars of the normal video processing pipeline are not explicitly covered. This processing is well known in the industry. However, in general, the normal video processing pipeline path will have YCbCr to RGB color space conversion and Gamma LUT correction before the end results are written to the RGB frame buffer. The failed (no-lock) video monitor and its associated processing path will be used to render the desired flagged failed video format when failed (no-lock) video is detected.

For full frame update rate performance of simultaneously displayed "n" analog video channels, the number of pipeline paths "p" would equal (mate directly to) the "n" video decoder/scalers available. This situation eliminates the need for the non-blocking switch network between the video decoder/scalers and the video processing pipeline paths. For designs that don't want/need to scale to that level of performance, a non-blocking switch network can be used to connect the larger amount of video decoder/scalers to the fewer amount of video processing pipelines as desired/needed. The behavior of this non-blocking switch network is such that any video decoder/scaler can be connected to any video processing pipeline path at any point in time. The pipeline dispatch/video frame complete path will command the done/available video processing pipelines to process the currently enabled video input streams. The currently enabled video input stream information is provided by the 'calculate individual video image size/location' process. This process also provides the necessary information to the individual video processing pipeline paths so correct placement of the rendered image will occur. The first done/available video processing pipeline gets the next enabled video input stream, the second the next, etc. This process continues until all currently enabled video input

[illegible] [redacted]

streams are processed. When all the currently enable video input streams are processed, the video frame is complete and the video frame buffer will now begin to refresh the display with that new information.

The video RGB frame buffer is the end results of the video processing activity. The video RGB frame buffer arrangement supports the simultaneous activities of receiving the multi-channel video processing results and sending those results to the RGB graphics display. A write interleaving multiplexer is used to write the results of the "p" video processing pipeline paths. The behavior of this multiplexer is that it can keep up in time (it has enough write data bandwidth) to successfully write of all of the results of the "p" video processing pipeline paths. By way of example, typically the video processing pipeline clock rate is twice the needed video processing write data clock rate to the video frame buffer when working with the standard 8-bit YCbCr data format from the decoder/scalers. This means (typically) that a clock multiplier network of "p/2" of the video processing clock rate (where "p" is the number of video processing pipelines used) is needed to drive the write interleaving multiplexer supporting the needed video RGB frame buffer write data clock rate (bandwidth). A video frame buffer write clock rate of 1x the video processing pipeline clock rate will handle 2 video processing pipelines. A video frame buffer write clock rate of 2x the video processing pipeline clock rate will handle 4 video processing pipelines, etc. The read out process from the video frame buffer to the display is accomplished at the needed display pixel clock rate. The read out pixel clock rate is a function of the display size and the desired display refresh rate. The frame buffer read out pixel clock rate may or may not be the same clock rate as the frame buffer write data clock rate.

<u>Robust Multi-Channel Real Time Video – Video Scaler Field Processing Enhancement for Unsynchronized Video Systems</u>

An additional improvement to the multi-channel real time video frame update rate (of "n" simultaneously display video sources) can be achieved when using a video scaler in an unsynchronized video system. Depending on the individual image size selected/calculated, it is possible to scale the needed 4:3 aspect ratio image output to receive all the needed image lines within a given (60Hz/50Hz) field. This allows the sampling and processing of incoming video to occur at twice the normal interlaced (30Hz/25Hz) frame rate which thereby can improve the overall video frame update rate in a multi-channel unsynchronized video system. A multi-channel synchronized video system will not see any improvements with this method since all video fields/frames are aligned in time and the faster field-sampling process simply will wait in time to complete all the selected images.

<u>Summary</u>

This scalable architecture will significantly enhance the display of multiple simultaneous analog video channels in real time. Up to full video frame update rates can be achieved and failed video sources are promptly flagged in real time. As few as two video processing pipelines can significantly improve the visual performance of a traditional "divide-by-n" type of degraded frame rate level of performance. The trade-off of logic gate counts (a potential limiting factor in the number of video processing pipelines) and memory bandwidth (to the video frame buffer) can easily be made in setting the desired

[illegible] [redacted]

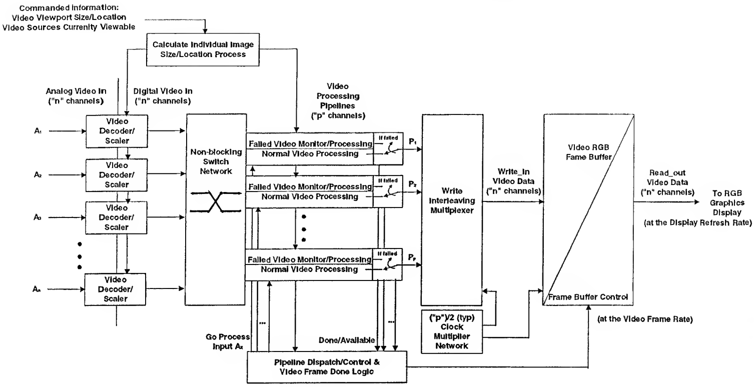
video frame update rate performance level. Supporting full frame rate performance for "n" channels of simultaneously displayed analog video can be achieved with an equal number of "p" pipelines.

The key aspects of this scalable method to robustly display multiple channels of real time analog video information are:

- Support scalability of multiple video processing pipelines to achieve up to full (60Hz/50Hz) analog video frame rate performance on a RGB graphics display when displaying "n" video channels simultaneously. This entails:
  - To eliminate the (dead) time associated with locking onto the analog video signal it is highly recommended that "n" video decoders are present to support the desired "n" channels of simultaneously displayed video.
  - From commanded user input of video viewport window size/location and the number of video source currently viewable, dynamically calculate the individual image size/location. This information is used by the video scalers and the video processing pipelines to dynamically support changing the number of viewable video images and their respective sizes.
  - Sizing the number of video processing pipelines with logic gates counts available (up to supporting "n" pipelines for full frame rate performance). A non-blocking switch network is needed to connect the "n" input video channels to the "p" video processing pipelines (when "n" is not equal to "p").
  - Sizing the video frame buffer write bandwidth to keep up in time with the number of "p" video processing pipelines used. A write interleaving multiplexer running the appropriately scaled video processing system clock will accomplish this task. A "p/2" clock multiplier network is typically needed where "p" is the number of pipelines implemented.
  - Pipeline dispatch/control and video frame done logic manages the multiple processing pipelines in time. This scales as needed to support the desired multiple processing pipelines as implemented.
- Supporting the ability to promptly detect and display individual failed video sources along with a method to promptly recover a 'now' passing video source.
- Support for improved unsynchronized video system update rates by using video-scaling field processing for image sizes that will support that method in real time.

[illegible] [redacted]

A Scalable Method to Robustly Display Multiple Channels of Real Time Synchronized or Unsynchronized Analog Video Information at up to Full Video Frame Rate Performance on a RGB Graphics Display



Commanded Information:
Video Viewport Size/Location
Video Sources Currently Viewable
Calculate Individual Image Size/Location Process
Analog Video In ("n" channels)
Digital Video In ("n" channels)
A₁
A₂
A₃
Aₙ
Video Decoder/ Scaler
Non-blocking Switch Network
Video Processing Pipelines ("p" channels)
Failed Video Monitor/Processing
Normal Video Processing
If failed
P₁
P₂
Pₚ
Write Interleaving Multiplexer
("p")/2 (typ) Clock Multiplier Network
Write_In Video Data ("n" channels)
Video RGB Fame Buffer
Frame Buffer Control
Read_out Video Data ("n" channels)
To RGB Graphics Display (at the Display Refresh Rate)
(at the Video Frame Rate)
Go Process Input Aₓ
Done/Available
Pipeline Dispatch/Control & Video Frame Done Logic

[illegible] [redacted]

APPENDIX B

--
-- Project Name : MAU CIO/Video Module
-- Used On : Video Module FPGA
--
-- File Name : vm_top15xx.vhd
--
-- Associated Files
-- package : none
-- entity : vm_top
-- architecture : rtl
--
-- Description: This is the top level module that ties all other modules together.
-- More detailed descriptions of the functions included in this block are included with the signal descriptions below.
--
-- Parking lot issues (J.Fye comments, last update[redacted]):

-- 1.) Support of PAL camera type vs. NTSC (ver '1223' changes), s/b resolved but still need to fully verify PAL format in lab...
-- Noticed in the lab that the SAA7112 decoder setting SA08 of 89h (D1&D0 = 01, searching mode) won't lock onto a PAL
-- input unless the VM is powered up into a PAL camera senerio. It could go from PAL (on power up) to NTSC but not vv.
-- Howevcr, the SAA7112 decoder setting SA08 of 88h (D1&D0 = 00, normal mode) does switch between NTSC/PAL at will.
-- However that setting does not handle aysnc cameras. Therefore it appears that to switch from NTSC to PAL we will
-- need to 'update' with SA08 at 88h and then do a second update with SA08 at 89h. The scaler paramcters will also
-- need to be changcd between thc two formats in order to 'correctly' see the picture since NTSC is 240 interlaccd
-- lines of active video while PAL is 288 interlaced lines of active video. Therefore, the VM SW nccds to know if it's
-- doing PAL and the VM SW must use the USER DEFINED image sizc selection option (selection 7 in HRD Tablc 9) to
-- accommodate thc PAL cameras. It is rccommended that the APM (the A/C Application Personality Modulc) the A/C
-- camera type info. (NTSC or PAL) which is made available to the VM SW. Note that thc VM cannot mix NTSC/PAL cameras

```
--     (it's either NTSC or PAL, not both).  The VM HW built-in image size selections (selections 0 thru 6
in HRD Table 9)
--     use NTSC scale factors.  The VM HRD has sample PAL settings for the decoder/scaler device which
VM SW can use in
--     their product.
-- 2.) There is (was) no logic to preclude the video image from going beyond the commanded output
lines... This has the
--     potential effect of going beyound the specified max 600 lines (i.e. lines 0 through 599) & into the
three specified
--     test lines (i.e. lines 600, 601 & 602).  If extra video lines do come in & corrupt the test lines, then
logic needs
--     to be added to not allow this.  This logic was added to version '1503' & of course retained for the
'1520' (& beyond)
--     versions.  However, version '1224' does not have this logic but it appears that the SAA7112 device
did not need it
--     while the SAA7115 did... The SAA7115 would routinely overrun it's commanded output lines when
sync'ing up to the
--     video input & hence it seems it could easily happen during normal operation.
--
-----------------------------------------------------------------------------------------------------------------


-- Modification History :
--
-- Sim file changes... when done put back the following 800x600 (landscape) initial conditions:
--  cr_memory15.vhd... int_image_sel_reg <= "00000001"; --at pwr-up w/1 camera selected (1=selected)
--    internal_sourc_sel_reg <= "110"; --at pwr-up select ROM data for image size select "6".(1x1,
landscape, frame processing).
--    frame_skip(=1), x_start(=0), y_start(=0), x_end(=799), y_end (=599),
--    x_dumax(=799), & y_dumax(=599).
--  scalerC_ROM15.vhd... cam_pixels = 300, cam_fieldlines = 112 (for 300x224 images).
--  ibc_addr_gen.vhd... Hblank clks(=19, yields 21 clks low)
--  i2c_serial_data15.vhd... gate_160(=161) & gate_10(=10)
--  i2c_serial_data4clkchip.vhd... gate_160(=161) & gate_10(=10)
--  red_x_buff.vhd... x_width=8 (pixel line width)
    -- read_buff.vhd... BLANK_DELAY_COUNT <= "1100100010110"; -- set for 6422 (x1916) clks
    (200us) {set to 2 clks for sim}
--  resetlg.vhd... CONSTANT two_ms := x"FADC"; -- 2ms count (set to 4 clk count for sim)
--  vm_top15xx.ucf... set block RAMs to zeros (set to linear patterns {except as noted} for sim)
-- Author     | Mod. Date | Change Made
-- Jim Fye    | [redacted] | Created from ver "1224" (TT7033542-104) virtex_vm_top12.vhd. This is VM
FPGA ver "1503".
--                      Use the Synplicity project file virtex_vm_fpga_top15.prj to compile this version.
--                      1.) First difference is that the SAA7115's I2C address is 42h while the SAA7112's
I2C
--                         address is 40h.
--                           Created file: i2c_serial_data15.vhd
--                      2.) Second difference is that the FPGA version is highlighed as "15xx" to represent a
--                         compile for supporting the SAA7115 Decoder/Scaler. The seperate compile is
needed
--                         since the SAA7115's I2C address is 42h while the SAA7112's I2C address is 40h &
there
--                         are SA register changes.
--                           Created file: cr_memory15.vhd
--                      2.) Third difference is that the SAA7115 image port control/data signals 'idq/ip_d'
occur
--                         concurrently with control signals igpv/igph. This required a change to the
ip_control
--                         state machine.
--                           Created file: ip_control15.vhd
--                      3.) To support the 7115 SA register changes the following files were created. The
"12xx"
--                         version supported the following write sequence to the SAA7112 device:
--                           SA 02 thru 17 / SA 80 thru 88 / SA 90 thru BF / SA 80 (reset release)
--                         The "15xx" version supports the following required SAA7115 write sequence:
--                           SA 01 thru 1D / SA 80 thru 88 / SA 90 thru BF / SA 88 (reset release)
--                         In addition, some bit definitions changed between the device's SA register set...
--                           Created files: adr_dat_mux15.vhd, decoderA_rom15.vhd, scalerA_rom15.vhd,
--                                     scalerB_rom15.vhd, scalerC_rom15.vhd, scalerD_rom15.vhd,
--                                     scalerE_rom15.vhd, scalerF_rom15.vhd, scalerG_rom15.vhd,
--                      4.) Created 'ycam_full' signal to stop writing to lines once the expected lines have
been
--                         received. This will ensure that a decoder/scaler line over run won't corrupt test
--                         lines. The SAA7112 did not seem to overrun it's expected line count, however, the
--                         SAA7115 seemed to right after an 'update' sequence for at least one frame but then
--                         would then stop & behave as expected. The 'ycam_full' signal is used at the top
level
--                         to block the ping/pong write strobe.
--                           Changed files: ibc_addr_gen.vhd, ibc_top.vhd, virtex_vm_top15.vhd
-- Jim Fye     |[redacted]     | Release version "1520" (baseline -1903 CCA FPGA version '1520' from -1902
CCA FPGA version
--                        '1503'/'1224')
--                          Renamed file: virtex_vm_top15.vhd to vm_top15xx.vhd
--                          Changed file: cr_memory15.vhd
--                        1.) Add support for the dedicated i2c interfaces that now connects to each Video
Decoder/Scaler
--                          device (qty of 8).  SCLK & SDATA signals have become SCLK[1:8] &
SDATA[1:8].  The SCLK1
--                          & SDATA1 interface has the Video Encoder device while SCLK[2:8] &
SDATA[2:8] do not.
--                          The same Video Decoder/Scaler write data is broadcasted on all eight interfaces.
The
--                          Video Encoder write data is only sent on interface #1.
--                            Changed files: vm_top15xx.vhd, host_if.vhd, i2c_if_top.vhd,
i2c_serial_data15.vhd,
--                              i2c_if_sm.vhd
--                        2.) Add support for writing to the SDRAM clock buffer chip used on the 7024377-
1903 version
--                          of the Video Module CCA.  The SDRAM clock buffer chip (CY2318ANZ &/or
W40S01-04) uses
--                          an SMBus (I2C) interface.  New signals CK_DATA & CK_CLK were added.
--                            Changed files: vm_top15xx.vhd, i2c_serial_data4clkchip.vhd (new)
--                        3.) Add support for local VM reset logic (output signal VM_RSTn) & internal FPGA
reset
--                          logic (signal INT_RSTn).  These are asserted asynchronously with input reset
signals
--                          RST_SYSn or SYS_RSTn.  These are de-asserted synchronously ~2ms after reset
inputs
--                          RST_SYSn or SYS_RSTn are de-asserted.  However, the RST_SYSn path
(HBM_RST# or P_V)
--                          will actually re-program the FPGA.
--                            Changed files: vm_top15xx.vhd, resetlg.vhd (new)
--                        4.) Add support for ZBT (NoBL) Frame Buffer interface.
--                            Changed files: vm_top15xx.vhd, zbtif.vhd (new), u_zero_buff.vhd, read_buff.vhd
--                        5.) Add support for direct connection of eight decoder/scalers.
--                          Another camera_processing_pipeline was added (i.e. a second instance of
ycbcr2rgb &
--                          red_x_buff) which added 798 slices (240 slices for ycbcr2rgb & 558 slices for
--                          red_x_buff).  The "1520" baseline slice count (from the above changes but prior to
the
--                          addition of the second camera_processing_pipeline) was ~3350 slices.  Ideally we
would
--                          match the VM Frame Buffer Write Bandwidth with four
camera_processing_pipelines.
```

```
--                        However, with the current cost constraints of "do the best you can do with an
XCV400E
--                        device" we seem to have the following options:
--                         One camera_processing_pipeline(32Mhz or 64Mhz FBi/f) = ~3400 slices (fit in
XCV400E's 4800 slices)
--                         Two camera_processing_pipelines(32Mhz or 64Mhz FBi/f)= ~4200 slices (fit in
XCV400E's 4800 slices)
--                         Three camera_processing_pipelines(64Mhz only FBi/f)  = ~5000 slices (fit in
XCV600E's 6912 slices)
--                         Four camera_processing_pipelines(64Mhz only FBi/f)   = ~5800 slices (fit in
XCV600E's 6912 slices)
--                        It could be that 200 slices could be absorbed into "slices containing unreleated
logic"
--                        for the "three camera_processing_pipelines(64Mhz FBi/f)" option to maintain the
desing
--                        in the XCV400E device, but decided to run with the "two
camera_processing_pipelines(32Mhz FBi/f)"
--                        design... going with the lower power design (32Mhz FBi/f) option since gate count
too
--                        close.  This needed to be decided up front since a two clock domain design (32Mhz
cam
--                        core with 64Mhz FB i/f) is architectually different from a single clock domain
design
--                        (32Mhz cam core with 32Mhz FB i/f).
--                          Changed files: vm_top15xx.vhd, ycbcr2rgb_top1.vhd, ip_control15.vhd,
ibc_top.vhd,
--                            write_buff.vhd, ibc_addr_gen.vhd, zbtif.vhd, camera_select.vhd,
cr_memory15.vhd,
--                            host_if.vhd
--                      6.) Take advantage of faster update possibilities when running async cameras/video
and
--                        running field processing (vs. frame processing).  Changed the multi-pipeline
camera-
--                        select-state-machine to move-on after the even field is captured (FID=0) and not to
--                        wait for the odd field (FID=1) to complete since it's thrown away anyhow.  Of
course
--                        for sync cameras/video there will be no performance improvement possibilities due
to
--                        this update.
--                          Changed files: vm_top15xx.vhd, camera_select.vhd
--                      7.) Created 'xcam_full' signal to stop writing to pixels once the expected pixel count
has
--                        been received.  Noticed in simulations that the first line after an async start into
--                        the multi_camera_proccssing_pipeline design that occasionally an extra pixel is
written
--                        to.  This could create a dangling pixel if it happens on the last camera on the line.
--                   Therefore to preclude that from happening, 'xcam_full' logic was created which is
--                   similar to the 'ycam_full' logic.
--                     Changed files: ibc_addr_gen.vhd, ibc_top.vhd, vm_top15xx.vhd
--                 8.) Since we have 8 dedicated Decoder/Scalers lock should be always valid (i.e.
locked) if
--                   the video input is valid/connected.  This allows for an immediate flagging of a
BAD_CAM
--                   input so no time is wasted to move on to the next video input.  In addition, since we
--                   now have dedicated decoder/scalers the time_out signal from the camera select
process
--                   may now be bypassed if no locked cameras are present... We will still sync to
--                   pipeline1's SOF input (for optimum synchronized performance), however, we will
bring
--                   the time_out process into the read out process and have it be fixed at 32.65mS if
SOF
--                   is not present.
--                     Changed files: camera_select.vhd, read_buff.vhd, ibc_top.vhd, video15xx.vhd
--                 9.) Tried a test FPGA PnR with the zbtif.vhd (both PING & PONG) running at NET
"C_64MHz"
--                   PERIOD = 13 ns HIGH 50 %; # derate: 64.11Mhz*1.2=76.932Mhz... With an PnR
effort level
--                   5 selected, all timing constraints were met except for the following:
--                   --------------------------------------------------------------------------------
--                     Constraint                         | Requested  | Actual    | Logic
--                                                    |          |           | Levels
--                   * TS_32_TO_64 = MAXDELAY FROM TIMEGRP "C_32 | 12.000ns   | 12.037ns
| 6
--                     MHz" TO TIMEGRP "C_64MHz" 12 nS          |          |          |
--                   --------------------------------------------------------------------------------
--                     TS_64_TO_32 = MAXDELAY FROM TIMEGRP "C_64 |            |          |
--                     MHz" TO TIMEGRP "C_32MHz" 12 nS          |          |          |
--                   --------------------------------------------------------------------------------
--                   The failing constraint represents the MAXDELAY for logic that goes from the
32Mhz
--                   domain to the 64Mhz domain and probably could be achievable (within the slowest
-6
--                   speed grade) with a multi-pass PnR run.  The other path indicates that the design
--                   currently does not have logic going from the 64Mhz domain to the 32Mhz domain
(which is
--                   true as the frame buffer data currently propagates through to the RIB TX data
outputs
--                   and gets latched for CPU reads in the "C_25MHz_F" domian).  HOWEVER, still
concerned
--                   about a 'MINDELAY' path possibility that falls within the +/-1ns skew potential
between
```

```
--                the 32Mhz & 64Mhz clocks...  In Xilinx's timing analyzer with only TS_32_TO_64
selected
--                & set to 'min' (vs -6, -7, -8 'max' settings) & set to show all paths (no limits) the
--                above slowest path goes to 4.244ns (1.603ns logic, 2.641ns route/37.8% logic,
62.2% route)
--                while the fastest path goes to 0.937ns (0.718ns logic, 0.219ns route/76.6% logic,
23.4% route).
--                Since the fastest 'min' path is less than 1ns we could have a race condition which
--                could give us a logic errors.  I suppose we could play some tricks with using the
--                falling edges of clocks in the control path at the expense of raising the
TS_32_TO_64
--                & TS_64_TO_32 performance bars to 13ns/2-1ns=5.5ns which is in the relm of
possibilities
--                for a control path signal... HOWEVER we seem to be out of FPGA gates... this
route had:
--                 Design Summary:
--                   Number of errors:      0
--                   Number of warnings:   13
--                   Number of Slices:           4,798 out of  4,800   99%
--                   Number of Slices containing
--                     unrelated logic:            28 out of  4,798    1%
--                   Number of Slice Flip Flops:   2,893 out of  9,600   30%
--                   Total Number 4 input LUTs:    7,872 out of  9,600   82%
--                     Number used as LUTs:                 6,135
--                     Number used as a route-thru:           232
--                     Number used for Dual Port RAMs:          1,056
--                     (Two LUTs used per Dual Port RAM)
--                     Number used as 16x1 ROMs:              436
--                     Number used as Shift registers:         13
--                   Number of bonded IOBs:        371 out of    404   91%
--                     IOB Flip Flops:                  450
--                   Number of Tbufs:              711 out of  4,960   14%
--                   Number of GCLKs:                4 out of      4  100%
--                   Number of GCLKIOBs:             4 out of      4  100%
--                So we've gone over 100% by using "Slices containing unrelated logic" which is hit
or
--                miss wrt how it actually fits a design... So I'd say we're done!
--                 Changed files: vm_top15xx.vhd, vm_top15xx.ucf
--             10.) There was a desire to support an "external RIB loopback test" such that test
engineering
--                could loopback one of the RIB TX outputs to the RIB RX input and verify
successful
--                transmissions.  This primarily would of been used to verify the RIB TXers and RIB
RXer.
--                It was envisioned that a software setable "loopback" bit would of been added to the
--                  VM register set to support this process & the process would of looked roughly as
follows:
--                  a.) With the "freeze" bit set (throughout each of these steps), SW fills the PING
--                      buffer ("ping1pong0" = 1) with a test pattern... while the PONG buffer is being
--                      countinously transmitted out the RIB.
--                  b.) SW sets "ping1pong0" = 0 which now swaps the buffers so that the PING
buffer is
--                      now being countinously transmitted out the RIB.
--                  c.) SW sets "loopback" = 1 which now synchronizes to the RIB RX data stream
and dumps
--                      the data into the PONG buffer (with "ping1pong0" = 0).
--                  d.) SW delays a few fields... say at least 3x60Hz or 50ms.
--                  e.) SW sets "loopback" = 0 which stops dumping the RIB RX data stream into the
PONG
--                      buffer.
--                  f.) SW verifies that the correct pattern is in the PONG buffer.
--                  HOWEVER, since the design (in the XCV400E device) is out of gates this support
was
--                  deferred.
```